SELF-REGULARIZED GUMBEL SIGMOID FOR COST-AWARE SUBSET SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Subset selection is a core problem in machine learning, but selecting useful features under heterogeneous candidate cost constraints remains challenging. Differentiable relaxations make it tractable, but many existing methods require tuning an explicit sparsity penalty and treat candidates as equally costly, limiting their applicability when resource costs vary. We introduce the Self-regularized Gumbel Sigmoid (SrGS), a simple framework that replaces penalty-based sparsification with a three-stage structural mechanism: a competitive normalizer over candidates; explicit cost encoding, which rescales the attention weight into retention probabilities of candidates that meet a budget expressed in physical resources such as memory or latency; and stochastic gating. SrGS adds only one learnable scalar per candidate and masks inputs without modifying the backbone network. We show that the default Softmax normalizer exponentially suppresses the gradients of losing candidates, replace it with scale-invariant Absolute Linear Normalization (ALN) and introduce a Utility-Aware Subgradient (UAS) to preserve optimization signals at its singularity, together with clipping that maintains gradient flow for suppressed candidates. For linear models with squared loss, we characterize the induced implicit regularizer and derive the budget allocation that minimizes it for fixed effective coefficients. The minimized regularizer vanishes exactly when the weighted feature support fits within the budget. Across five feature-selection benchmarks spanning click-through-rate prediction (CTR), and several high-dimensional speech, text, image datasets, we evaluate feature selection and cost-aware embedding dimension optimization under a common select-retrain protocol. Across these settings, SrGS is competitive with or outperforms representative greedy, pruning, and differentiable baselines, suggesting that a low-overhead selection mechanism can handle diverse domains and resource constraints without modifying the backbone.

1 INTRODUCTION

Subset selection underlies feature ranking, interpretability, and efficient architecture design. It is a first-order concern in large-scale recommendation and click-through-rate (CTR) prediction, where a single categorical field can occupy a multi-gigabyte embedding table, and it guards against overfitting in high-dimensional domains such as genomics. Selecting the best subset of a given size is NP-hard (Natarajan, 1995), and heterogeneous costs make it harder: the cardinality constraint becomes a knapsack constraint, so a selector can no longer simply keep the top K candidates. Even for monotone submodular utilities, where greedy selection is $(1 - 1/e)$-optimal under a cardinality constraint (Nemhauser et al., 1978), greedy under a knapsack constraint admits no constant-factor guarantee without partial enumeration (Khuller et al., 1999; Sviridenko, 2004), and neural utilities are not submodular. Practical methods therefore resort to continuous relaxations.

Existing methods, such as Dropout Feature Ranking (DFR) (Chang et al., 2017) and related relaxations, learn an input mask whose entries $z_j \in [0, 1]$ are retention probabilities, and control sparsity with an explicit penalty:

$$\mathcal{L}(\theta, z) = -\frac{1}{M}\sum_{i=1}^{M} \log p\big(y_i \mid f(x_i \odot z_i; \theta)\big) + \frac{\lambda}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} z_{ij}, \tag{1}$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

where M is the batch size, N the number of features, and z_{ij} the stochastic gate of feature j on sample i. This formulation has three limitations. (i) *Tuning:* λ must be found by grid search; too small a value yields no sparsity, too large a value collapses accuracy. (ii) *The vanishing gradient trap:* the penalty drives suppressed gates toward zero, and under common parameterizations such as Softmax (Section 4.1) their gradients shrink with them, so a feature down-weighted early is hard to recover even if it would be valuable in combination with features selected later. (iii) *Uniform pricing:* every candidate is charged the same, yet a high-cardinality embedding table can cost orders of magnitude more memory than a bucketized scalar, so a feature count is a poor proxy for the resource constraint that actually binds.

We introduce the Self-regularized Gumbel Sigmoid (SrGS), which removes the penalty and recasts selection as structural budget allocation: candidates compete for a single shared budget expressed in physical resources such as memory or latency. Our contributions are:

- *The SrGS framework:* SrGS is a simple framework for cost-aware subset selection. It adds only one real-valued logit per feature to mask the input, resulting in a low-overhead architecture, and requires no modification to the backbone network. The budget is specified directly in physical resources rather than through a penalty weight. (Section 3)
- *Gradient-preserving optimization:* we show that Softmax normalization exponentially suppresses the gradients of losing candidates, replace it with scale-invariant Absolute Linear Normalization (ALN), and address the resulting singularity at the origin with a Utility-Aware Subgradient (UAS). (Section 4)
- *Implicit regularization:* we give a second-order expansion of the regularization induced by mask sampling and characterize it exactly for linear models with squared loss in the hard-gate limit. For fixed effective coefficients, we derive in closed form the allocation of the budget across features that minimizes this regularizer. (Section 5)
- *Empirical validation:* on five feature-selection benchmarks and cost-aware embedding dimension optimization, SrGS is competitive with or outperforms seven feature-selection and three embedding-allocation baselines under a five-seed select-retrain protocol. (Section 6)

2 Related Work

Continuous relaxations for subset selection. Convex relaxations such as the ℓ_1 penalty (Tibshirani, 1996) were the first route to tractable sparsity. In deep learning, the Gumbel-Softmax (Concrete) trick (Jang et al., 2017; Maddison et al., 2017) enables backpropagation through discrete masks; Concrete Dropout (Gal et al., 2017) learns dropout rates as variational parameters, and DFR (Chang et al., 2017) uses them to rank features. LassoNet (Lemhadri et al., 2021) instead ties nonlinear first-layer weights to a sparse linear skip connection through hierarchical proximal updates. The selection methods among these control sparsity through an explicit penalty weight.

Sparse attention and greedy selection. Softmax assigns non-zero mass to every candidate and so cannot by itself produce an exactly sparse subset. Sparsemax (Martins & Astudillo, 2016) and α-entmax (Peters et al., 2019) drive low-scoring entries to exactly zero, while greedy selectors—Sequential Attention (Yasuda et al., 2023), OMP (Pati et al., 1993), Sequential Lasso (Luo & Chen, 2014), and batch-wise attenuation (Liao et al., 2021)—commit to features iteratively. Both families share a recoverability problem, a form of the vanishing gradient trap: zero-probability candidates receive no gradient, and greedy commitments are never revisited. SrGS takes the other side of this trade-off. It keeps a dense normalizer, so no candidate receives exactly zero probability, and obtains binary masks downstream through budget-scaled clipping and Straight-Through sampling (Bengio et al., 2013).

Related work on embedding dimension search and implicit regularization is discussed in Appendix B.

3 Our Framework: The Self-regularized Gumbel Sigmoid

SrGS eliminates the static penalty weight λ in favor of a three-stage structural process. Rather than asking the optimizer to trade task loss against a sparsity penalty, we constrain it to *allocate* a fixed budget of capacity. The complete algorithm is provided in the appendix (Algorithm 1).

Stage 1: Global competition. Each of the N candidate features carries a learnable, unconstrained score $t_j \in \mathbb{R}$. A normalizer $\mathcal{N} : \mathbb{R}^N \to \Delta^{N-1}$ maps the score vector to a competitive attention distribution $s = \mathcal{N}(t)$, so that $s_j \geq 0$ and $\sum_{j=1}^{N} s_j = 1$. The simplex constraint is what makes the stage competitive: increasing one feature's share necessarily decreases the others'. The default choice, inherited from attention mechanisms and from prior continuous subset selection methods, is the Softmax,

$$s_j = \frac{\exp(t_j)}{\sum_{k=1}^{N} \exp(t_k)}. \tag{2}$$

We adopt Eq. (2) for exposition here, but Section 4.1 shows that it suppresses the gradients of low-scoring candidates, and Section 4.2 replaces it with Absolute Linear Normalization. Stages 2 and 3 are unaffected by that substitution.

Stage 2: Explicit cost encoding and budgeting. Let $c_j > 0$ denote the physical cost of feature j (memory footprint, serving latency, or acquisition cost) and let B be the total cost budget. We convert the attention distribution into retention probabilities by rescaling it so that the expected cost of the sampled subset matches the budget:

$$\tilde{z}_j = \frac{s_j B}{\sum_{i=1}^{N} s_i c_i}, \qquad z_j = \mathrm{Clip}(\tilde{z}_j,\ \epsilon,\ 1-\epsilon). \tag{3}$$

The normalizing constant $\bar{c} = \sum_i s_i c_i$ is the attention-weighted mean cost, so before clipping the budget is met *exactly* rather than approximately: $\sum_{j=1}^{N} \tilde{z}_j c_j = B$.

Cost therefore acts collectively rather than per-feature: a feature does not receive an individual $1/c_j$ discount, but raising the attention share of an expensive feature inflates $\bar{c}$ and contracts every retention probability, so the optimizer pays for expensive choices out of the shared budget. After clipping, $\sum_j z_j c_j \leq B + \epsilon \sum_j c_j$, with equality to B whenever no clip is active; the small constant ϵ keeps the logits in Stage 3 finite; the lower bound is implemented with a straight-through estimator (Eq. 8, Section 4.3). Because $z_j \geq \epsilon$, SrGS never assigns a feature exactly zero probability, and sparsity emerges stochastically rather than by hard truncation.

Stage 3: Stochastic selection. We propose two approaches here. The first is the soft ($\tau > 0$) regime. Given the retention probabilities, we draw a binary mask with the Gumbel-Sigmoid estimator. Let $G_1, G_2 \sim \mathrm{Gumbel}(0,1)$ be independent and let $\tau > 0$ be the temperature:

$$\hat{w}_j = \sigma\left(\frac{\mathrm{logit}(z_j) + G_1 - G_2}{\tau}\right) \in (0,1), \qquad \mathrm{logit}(z) = \log z - \log(1-z). \tag{4}$$

The second is the hard ($\tau \to 0$) limit. Because a difference of two independent Gumbel variates is Logistic, the hard limit of Eq. (4) reduces to inverse-transform Bernoulli sampling,

$$w_j = \mathbf{1}\left[u_j \leq z_j\right], \qquad u_j \sim \mathrm{Uniform}(0,1), \tag{5}$$

which is the regime analyzed in Section 5. The hard limit removes the temperature hyperparameter, leaving the clip constant ϵ and the annealing schedule as the remaining settings. In this case, the mapping is discontinuous. To retain gradient flow through the discrete draw we apply the Straight-Through Estimator (Bengio et al., 2013):

$$m_j = z_j + \mathrm{sg}[w_j - z_j]. \tag{6}$$

Throughout, $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator. In the hard limit, the forward pass therefore sees the discrete mask $m_j = w_j \in \{0,1\}$, while the backward pass differentiates z_j. The masked input $\hat{X} = X \odot m$ is passed to the downstream model.

In our runs, the soft gate performed slightly better, so we use a $\tau = 0.1$ in all experiments.

Inference. At deployment no sampling is performed. We rank candidates by their converged retention probability z_j (equivalently, their learned logit magnitude $|t_j|$ under ALN) and sort candidates in decreasing order of $|t_j|$, $|t_{(1)}| \geq |t_{(2)}| \geq \cdots$, and scan this order starting from $S = \varnothing$. For each candidate j, add it to S if $\sum_{i \in S} c_i + c_j \leq B$; otherwise skip it and continue to the next candidate. After considering all candidates, return S. Therefore, this procedure guarantees that the total cost of the selected subset never exceeds the budget. In the uniform-cost setting ($c_j \equiv 1$, $B = K$) this is simply the top-K set.

4 ESCAPING GRADIENT TRAPS: FROM SOFTMAX TO UTILITY-AWARE ALN

4.1 THE SOFTMAX VANISHING GRADIENT TRAP

Continuous subset selection and attention mechanisms default to the Softmax of Eq. (2) for its smooth differentiability and strictly positive outputs. However, this comes at a cost: the logit gradient of a low-probability feature vanishes with its probability.

Let $\mathcal{L}$ be the network loss and let $g_i = \frac{\partial \mathcal{L}}{\partial s_i}$ be the *downstream utility* of feature i, so that a strongly *negative* g_i marks a feature whose presence reduces the loss. Let $\bar{g} = \sum_{k=1}^{N} s_k g_k$ be its attention-weighted mean, and define the Utility Advantage $U_i = g_i - \bar{g}$. Because $\bar{g}$ is a convex combination of the g_k, the advantages straddle zero: a feature with $U_i > 0$ is losing the competition and is pushed down, while one with $U_i < 0$ is pushed up, irrespective of the absolute magnitude of its gradient. With the Softmax Jacobian $\frac{\partial s_j}{\partial t_i} = s_i(\delta_{ij} - s_j)$, the chain rule gives

$$\frac{\partial \mathcal{L}}{\partial t_i} = \sum_{j=1}^{N} g_j \frac{\partial s_j}{\partial t_i} = s_i \cdot \left(g_i - \sum_{j=1}^{N} s_j g_j \right) = s_i \cdot U_i. \tag{7}$$

Here $-U_i$ is the baseline-subtracted advantage of a REINFORCE policy gradient (Section 7). Since $s_i < e^{t_i} / \sum_{j \neq i} e^{t_j}$, we have $s_i = O(e^{t_i})$ as $t_i \to -\infty$; for bounded g, $|U_i|$ is bounded and hence $\left| \frac{\partial \mathcal{L}}{\partial t_i} \right| = O(e^{t_i}) \to 0$. A gradient step moves t_i by $O(\eta e^{t_i})$, so the number of steps required to raise t_i by a fixed increment grows exponentially as t_i decreases. Once its score has collapsed, a feature that later becomes valuable ($U_i \ll 0$), e.g., as the network adapts, receives only an exponentially small update.

This situation is common: in a competitive allocation with unequal utilities, below-average features exist at every training step, and gradient descent pushes their logits downward.

Lemma 1. *Consider full-batch gradient descent on all logits, starting from finite logits, with a common step size $\eta > 0$ and uniformly bounded downstream utilities. Unless all utilities coincide, some feature has $U_j^{(n)} > 0$ at each step n. For any fixed feature i with $U_i^{(n)} > 0$ throughout, its logit decreases strictly: $t_i^{(n+1)} = t_i^{(n)} - \eta s_i^{(n)} U_i^{(n)} < t_i^{(n)}$.*

The proof is deferred to Appendix F.3. The lemma guarantees that *some* feature is below average at each step, not that it is the same one throughout, and it is stated for full-batch gradient descent, whereas our experiments use minibatch Adam, so the conclusion holds in expectation rather than stepwise. Still, it indicates that Softmax pushes persistently below-average features toward the region where their gradients vanish.

Nevertheless, the lemma also suggests how to avoid the Softmax gradient trap. The sign of the update is set by U_i alone; the difficulty lies in the magnitude factor s_i. In the next subsection, we therefore introduce Absolute Linear Normalization (ALN), which removes the s_i prefactor and its associated suppression, at the cost of relocating the difficulty to the origin $t_i = 0$.

4.2 SCALE-INVARIANT ABSOLUTE LINEAR NORMALIZATION

Definition 1. *For raw scores $t \in \mathbb{R}^N$, the ALN score is $s_i(t) = \frac{|t_i|}{Z}$, where $Z = \sum_{k=1}^{N} |t_k|$.*

ALN is homogeneous of degree 0, $s_i(c \cdot t) = s_i(t)$ for any scalar $c \neq 0$: rescaling the logits does not change the selection probabilities.

Lemma 2. *Under ALN, the logit gradient is $\frac{\partial \mathcal{L}}{\partial t_i} = \frac{\operatorname{sgn}(t_i)}{Z} U_i$ for $t_i \neq 0$. Thus the magnitude of the logit gradient $\left| \frac{\partial \mathcal{L}}{\partial t_i} \right|$ is independent of its own score s_i.*

The proof is deferred to Appendix F.6. The gradient still scales as $1/Z = 1/\|t\|_1$, which scale invariance leaves unconstrained. ALN thus replaces the per-feature exponential decay of Softmax by a factor shared by all features, which does not favor any feature and can be offset by the step size or by adaptive optimizers such as Adam. However, the signed gradient $\pm |U_i|/Z$ has a jump discontinuity at $t_i = 0$, and ALN actively pushes losing features toward it.

Lemma 3 (The ALN Origin Trap). *Let feature i satisfy $U_i > 0$ at every step of full-batch gradient descent under ALN with step size η, and let $a = \eta U_i/Z$. While $|t_i| > a$, each step reduces $|t_i|$ by a; once $|t_i| \le a$, the next iterate again satisfies $|t_i| \le a$. Thus t_i is driven toward the origin and then oscillates around it.*

The proof is deferred to Appendix F.4. ALN thus exchanges one degeneracy for another: Softmax drives losing features to $t_i \to -\infty$, whereas ALN keeps the gradient bounded away from zero (Lemma 2) but drives those features toward the single point where the normalizer is not differentiable. There, AD engines such as PyTorch and JAX define $\partial|t_i|/\partial t_i$ as hard-coded value such as 0 or 1, re-creating the vanishing gradient trap. Standard clipping remedies do not resolve this (Appendix F.5): forward-pass clamping (e.g., `torch.clamp`) has zero derivative on its inactive branch and turns the singularity into a dead-zone, while in-place projection $t \mapsto \max(t, \epsilon)$ breaks both the scale invariance and the sign symmetry of ALN, installing an absorbing boundary where suppressed features accumulate.

4.3 The Utility-Aware Subgradient (UAS) Resolution

Two mechanisms are required: one to stop the probability clip from creating a dead-zone, and one to stop the origin of the ALN map from becoming absorbing.

The Straight-Through Estimator clip. Since $\mathrm{logit}(p) = \log p - \log(1-p)$ diverges at $p \in \{0, 1\}$, the budget-scaled scores $\tilde{z}_i$ of Eq. (3) must be bounded away from $\{0, 1\}$ before the Gumbel gate. We treat the two bounds asymmetrically:

$$z_i := \min\Big(\tilde{z}_i + \mathrm{sg}\big[\max(\tilde{z}_i, \epsilon) - \tilde{z}_i\big],\ 1 - \epsilon\Big). \tag{8}$$

The lower bound is imposed with a straight-through estimator (Bengio et al., 2013), so $\partial z_i/\partial \tilde{z}_i = 1$ for $\tilde{z}_i < 1 - \epsilon$ and suppressed features keep their utility signal, avoiding the dead-zone of forward-pass clamping. The upper bound is a standard clip with $\partial z_i/\partial \tilde{z}_i = 0$ for $\tilde{z}_i > 1-\epsilon$. This asymmetry is deliberate: a saturated feature is already selected almost surely, and passing its gradient straight through would let a highly informative feature keep inflating its attention share, draining the simplex and lowering the retention probabilities of all other features without any gain in the forward pass. Under the standard clip, saturated features retain their full cost c_i while the remaining features compete for the residual budget, mirroring the saturated/unsaturated split of Theorem 2. A saturated feature can still be deselected, since its logit continues to receive gradient through the budget normalizer $\sum_k s_k c_k$ of the other features.

The Utility-Aware Subgradient. At the origin, $\partial|t_i|/\partial t_i$ is undefined. Since $|\cdot|$ is convex, every value in $[-1, 1]$ is an admissible subgradient at 0; UAS selects the one aligned with the direction the loss is pushing:

$$\left.\frac{\partial|t_i|}{\partial t_i}\right|_{t_i=0} \triangleq \mathrm{sgn}\left(\frac{\partial \mathcal{L}}{\partial |t_i|}\right) = \mathrm{sgn}(U_i) \ \in\ \{-1, +1\}, \tag{9}$$

which by Lemma 2 yields, at a dead parameter,

$$\left.\frac{\partial \mathcal{L}}{\partial t_i}\right|_{t_i=0} = \frac{\mathrm{sgn}(U_i)}{Z}\, U_i = \frac{|U_i|}{Z} > 0 \qquad \text{whenever } U_i \neq 0. \tag{10}$$

Unlike fixed conventions (a constant ± 1, a random draw from $[-1, 1]$, or the default 0, which recreates the trap of Lemma 3), this choice adapts to the feature's current Utility Advantage.

Since Eq. (10) is strictly positive, the update $t_i \leftarrow -\eta|U_i|/Z$ displaces a dead parameter into the negative domain, where Lemma 2 gives $\partial\mathcal{L}/\partial t_i = -U_i/Z$. If the feature is valuable ($U_i < 0$), descent drives t_i further negative and its score recovers, the negative domain being equivalent to the positive one under ALN. If it deserves suppression ($U_i > 0$), the parameter is pulled back and oscillates within a neighborhood of radius $O(\eta|U_i|/Z)$ of the origin. UAS thus restores a signal at the origin without preventing the suppression of unhelpful features.

5 THE IMPLICIT REGULARIZER OF SRGS

For fixed retention probabilities $z \in [0,1]^N$ and coefficients $\theta \in \mathbb{R}^N$, each SrGS gate uses its own independently drawn uniform variable $u_i \sim \mathrm{Uniform}(0,1)$. As $\tau \to 0$, the gates become $w_i = \mathbf{1}\{u_i \le z_i\}$, so the masks are independent Bernoulli samples with parameters z_i. The mean masked (effective) coefficients are $\beta := z \circ \theta$, where $\circ$ denotes the elementwise product. For a smooth loss $\mathcal{L}$, a second-order Taylor expansion in the mask coordinates about z gives:

$$\mathbb{E}_w[\mathcal{L}(w \circ \theta)] \approx \mathcal{L}(z \circ \theta) + \frac{1}{2} \sum_{i,j} \frac{\partial^2 \mathcal{L}(z \circ \theta)}{\partial z_i \partial z_j} \mathrm{Cov}(w_i, w_j). \tag{11}$$

The first-order term vanishes because $\mathbb{E}[w_i - z_i] = 0$. For $i \neq j$, the independent sampling noises give $\mathrm{Cov}(w_i, w_j) = \mathbb{E}[(w_i - z_i)(w_j - z_j)] = 0$, whereas $\mathrm{Var}(w_i) = z_i(1 - z_i)$. These identities reduce the covariance sum to the diagonal terms. The derivatives are taken with respect to the mask coordinates of the composite function $z \mapsto \mathcal{L}(z \circ \theta)$.

To obtain an exact characterization, we focus on selecting features to approximate a target y with a linear model. Let $X \in \mathbb{R}^{n \times N}$ contain n observations of N candidate features, with feature columns X_j, and let $y \in \mathbb{R}^n$ be the target vector. For a sampled mask, the predictions are $X(w \circ \theta)$ and the squared loss is $\mathcal{L}(w \circ \theta) = \|y - X(w \circ \theta)\|_2^2$. The mean predictions are $X\beta = X(z \circ \theta)$, with loss $\mathcal{L}(\beta) = \|y - X\beta\|_2^2$. For feature costs $c_i > 0$, we analyze the idealized budget domain $z \in [0,1]^N$ with $\sum_i c_i z_i \le B$, where $B > 0$. This constrains the expected cost $\mathbb{E}_w[\sum_i c_i w_i]$.

Under this linear model, (11) separates exactly into the loss at the mean coefficients $\beta = z \circ \theta$ and a variance penalty induced by mask sampling, since the quadratic loss has no higher-order Taylor term, as follows: $\mathbb{E}_w[\mathcal{L}(w \circ \theta)] = \mathcal{L}(z \circ \theta) + R_{\mathrm{var}}(z, \theta)$, and $R_{\mathrm{var}}(z, \theta) = \sum_j \|X_j\|_2^2 \, \theta_j^2 \, z_j(1 - z_j)$. For fixed effective coefficients $\beta = z \circ \theta$, we minimize the variance penalty over alternative retention probabilities q and coefficients a satisfying $q \circ a = \beta$, under the same cost budget:

$$\mathcal{P}_{B,c} := \left\{ q \in [0,1]^N : \sum_i c_i q_i \le B \right\}, \; R(q; \beta) := \inf_{q \circ a = \beta} R_{\mathrm{var}}(q, a) = \sum_i \|X_i\|_2^2 \beta_i^2 \frac{1 - q_i}{q_i}. \tag{12}$$

Then the induced penalty under the cost budget is $R^*_{B,c}(\beta) := \inf_{q \in \mathcal{P}_{B,c}} R(q; \beta)$.

5.1 A WEIGHTED ℓ_0 CHARACTERIZATION

We first characterize when $R^*_{B,c}(\beta)$ vanishes. The following theorem shows that the zero set of the minimized masking penalty is exactly the set of effective coefficients $\beta = z \circ \theta$ whose active features fit within the cost budget.

Theorem 1. *Assume $c_i > 0$, $B > 0$, and $\|X_i\|_2 > 0$ for every feature. Define the weighted support cost of $\beta = z \circ \theta$ by $\|\beta\|_{0,c} := \sum_{i:\beta_i \neq 0} c_i$. Then we have $R^*_{B,c}(\beta) = 0$ if and only if $\|\beta\|_{0,c} \le B$.*

The proof is deferred to Appendix F.1.

5.2 CLOSED FORM: THE OPTIMAL FEATURE ALLOCATION

In this subsection, we give the closed form of $R^*_{B,c}(\beta)$ for $\beta = z \circ \theta$. Assume standardized columns $\|X_i\|_2 = 1$. For any index set A, write $\|\beta_A\|_{1,\sqrt{c}} := \sum_{i \in A} \sqrt{c_i} \, |\beta_i|$. Let $T = \mathrm{supp}(\beta)$.

Theorem 2. *If $\sum_{i \in T} c_i \le B$, then $R^*_{B,c}(\beta) = 0$. If $0 < B < \sum_{i \in T} c_i$, let $\lambda > 0$ be the scalar chosen so that the probabilities below exhaust the cost budget, $\sum_i c_i q_i^* = B$. Then, the optimal allocation is*

$$q_i^* = \min\left\{1, \frac{|\beta_i|}{\sqrt{\lambda c_i}}\right\}. \tag{13}$$

Let $S = \{i \in T : q_i^* = 1\}$, $U = T \setminus S$, and $b = B - \sum_{i \in S} c_i$. Then $U \neq \emptyset$, $b > 0$, and

$$R^*_{B,c}(\beta) = \frac{\|\beta_U\|_{1,\sqrt{c}}^2}{b} - \|\beta_U\|_2^2. \tag{14}$$

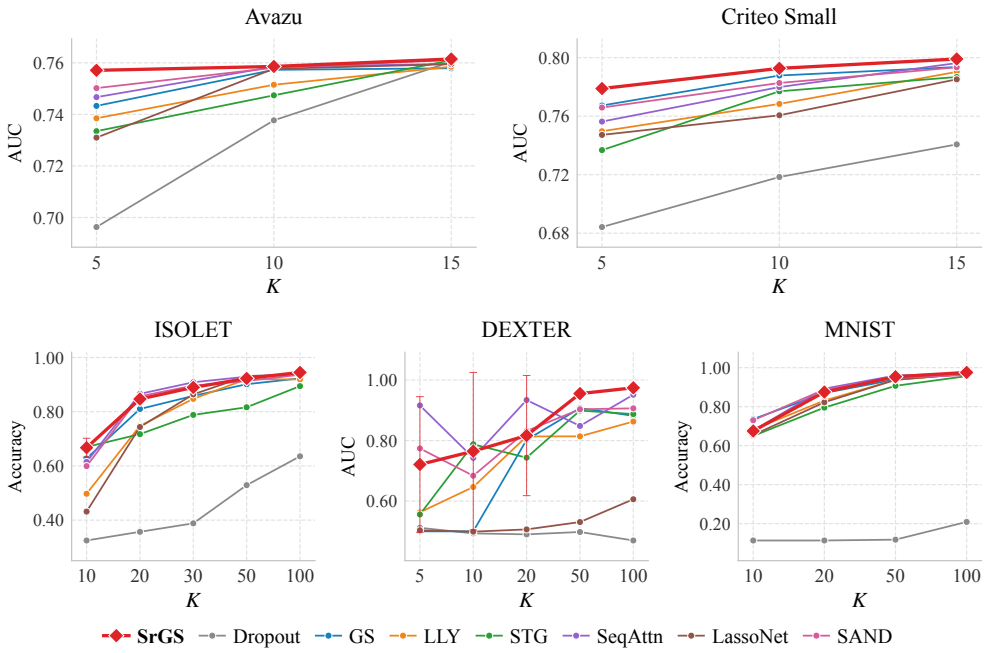

Figure 1: Type-1 feature selection on CTR datasets and high-dimensional datasets. Full numerical results, including losses, are in Tables 5 and 6, respectively.

The proof is deferred to Appendix F.2. The theorem implies that the penalty is a squared $\sqrt{c_i}$-weighted ℓ_1 norm with a negative ℓ_2 correction on the unsaturated support U. A saturated feature consumes its full cost c_i but contributes no variance penalty. The remaining features *compete* for $b = B - \sum_{i \in S} c_i$, and their probabilities scale as $|\beta_i|/\sqrt{c_i}$.

6 Experiments

In this section, we empirically evaluate the proposed SrGS framework against representative baselines in two subset-selection regimes: uniform-cost feature selection ($c_j \equiv 1$, Type-1), and cost-aware subset selection over candidate embedding blocks under heterogeneous parameter budgets ($c_{j,b} = 4V_j$, Type-2). We also perform an ablation study of normalization, annealing, and selection structure of SrGS.

6.1 Type-1: Feature Selection

Datasets. We evaluate five benchmarks: Avazu (Wang & Cukierski, 2014), Criteo (Criteo Labs, 2014), ISOLET[1], DEXTER[2], and MNIST[3]. Table 2 summarizes their features. The high-dimensional datasets are passed directly to the dense backbone; CTR fields are embedded with a uniform dimension $d = 20$ after the preprocessing described in Appendix C.1.

Baselines. We compare SrGS against seven representative feature selection methods, covering greedy attention, stochastic gates, and sparsity-regularized neural models: Gumbel-Softmax (GS; Jang et al., 2017; Maddison et al., 2017), Dropout Feature Ranking (Dropout; Chang et al., 2017), LLY (Liao et al., 2021), Sequential Attention (SeqAttn; Yasuda et al., 2023), Stochastic Gates (STG; Yamada et al., 2020), LassoNet (Lemhadri et al., 2021), and SAND (Pad et al., 2025). Detailed descriptions are provided in Appendix H.1.

[1] https://archive.ics.uci.edu/dataset/54/isolet
[2] https://archive.ics.uci.edu/dataset/168/dexter
[3] https://www.tensorflow.org/datasets/catalog/mnist

Protocol and Model Architecture. We adopt a two-phase **Select-Retrain Protocol** (Lemhadri et al., 2021):

Phase 1: Selection. We train the feature selection algorithm on the full feature set using an MLP. The high-dimensional datasets use an MLP with two fully connected hidden layers of widths $[128, 64]$. Avazu and Criteo use a three-layer MLP with widths $[768, 256, 128]$. Hidden layers use ReLU activations. The benchmark uses 50,000 training steps for the high-dimensional datasets and 300,000 for the CTR datasets. For SrGS, all features remain active for the first 10% of training; the allocation scale then decreases from N to K between 10% and 20% of training, followed by search at the target allocation scale. All methods use the same dataset splits and downstream backbone. Method-specific hyperparameters are tuned on validation performance and then fixed across the five evaluation seeds. Exact settings are provided in Appendix C.4.

Phase 2: Retraining. We instantiate a fresh model with the same architecture and train it using only the selected K features. This model is trained from scratch to ensure metrics reflect the subset quality rather than selection artifacts. We report classification accuracy for ISOLET and MNIST, AUC for DEXTER and the CTR datasets, and Retrain Test Loss (cross-entropy, lower is better) separately for CTR.

Results and Analysis. On Criteo and Avazu, SrGS achieves the highest mean AUC at all three budgets. It also attains the lowest Test Loss on Criteo, while SAND attains the lowest Test Loss on Avazu. The high-dimensional benchmarks show that the benefit of SrGS depends on both the dataset and the feature budget. At $K = 100$, SrGS outperforms every baseline in mean performance on all three datasets with small standard deviations. Under tighter feature budgets, some baselines achieve higher mean scores, but SrGS remains competitive in several settings when the reported variation across seeds is taken into account, as reflected by overlapping mean $\pm$ standard-deviation ranges. These extremely tight budget regimes also exhibit larger variation for several baselines, indicating that sensitivity to small feature budgets is not specific to SrGS.

Overall, SrGS is strongest beyond the most restrictive budgets.

6.2 Type-2: Cost-Aware Subset Selection for Embedding Dimension Optimization

Baselines. We compare cost-aware SrGS against three embedding allocation methods: Mixed Dimension Embeddings (MDE; Ginart et al., 2021), Plug-in Embedding Pruning (PEP; Liu et al., 2021), and i-Razor (Yao et al., 2023). Detailed descriptions are provided in Appendix H.2.

Protocol and Model Architecture. We use the Avazu and Criteo datasets, preprocessing, and CTR backbone of Type-1, and follow the same Select-Retrain protocol: we search for field dimensions under an embedding parameter budget and retrain a fresh model with the assigned dimensions. With V_j the vocabulary size of field j, the budget $B = \rho\, d_{\mathrm{ref}} \sum_{j=1}^{F} V_j$, with reference dimension $d_{\mathrm{ref}} = 20$ and ratios $\rho \in \{0.25, 0.50, 1.00, 1.50\}$, **counts embedding table parameters**. Each field is divided into 4-dimension blocks up to dimension 64 with the first block always active, so the $N = 15F$ optional blocks (j, b) carry heterogeneous costs $c_{j,b} = 4V_j$, and the selected dimensions $d_j \in \{4, 8, \ldots, 64\}$ satisfy $\sum_{j=1}^{F} V_j d_j \le B$. Further details are given in Appendix C.2.

All methods are evaluated under the same five seed Select–Retrain protocol. Exact optimization settings and the configurations of SrGS, i-Razor, MDE, and PEP are reported in Appendix C.4.

Results and Analysis. Figure 2 reports the main Type-2 AUC comparison. On Avazu, SrGS has the highest mean AUC at $\rho = 0.50$ and 1.00, and is within 10^{-5} of i-Razor at $\rho = 0.25$. At $\rho = 1.50$, PEP has a slightly higher mean AUC than SrGS, and PEP also attains the lowest loss at all four Avazu budgets. On Criteo, i-Razor attains the highest mean AUC at each budget, with SrGS trailing only slightly and the two methods' mean $\pm$ standard-deviation ranges overlapping at every budget. Overall, SrGS achieves the best or competitive performance across the evaluated settings, suggesting that essentially the same selection mechanism extends from uniform-cost feature selection to cost-aware subset selection.

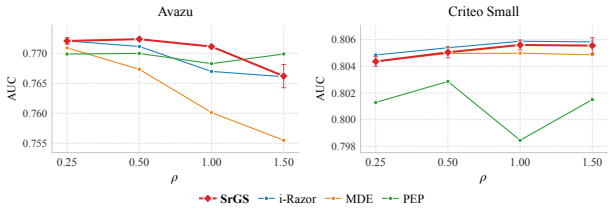

Figure 2: Cost-aware subset selection on Avazu and Criteo. Full numerical results are in Table 7.

Stages	Anneal	Softmax	ALN
1	Std.	$.9561 \pm .0029$	$.9535 \pm .0019$
1	Slow	$.9613 \pm .0011$	$\mathbf{.9642 \pm .0058}$
2	Std.	$.9595 \pm .0023$	$.9624 \pm .0037$
2	Slow	$.9629 \pm .0019$	$.9638 \pm .0009$

Table 1: Factorial ablation on MNIST with $K = 50$ (accuracy).

6.3 Ablation Study

In this subsection, we conduct an ablation study to evaluate the effects of SrGS components and several additional optimization ideas. We use the MNIST benchmark from Type-1 and focus on a single feature budget of $K = 50$. We consider three variants of SrGS and evaluate their combinations, retraining each configuration for 50,000 steps over five seeds. **(1) Two-stage selection:** We first retain 200 candidates and then select 50 features from this reduced set, instead of selecting 50 features directly in one stage. **(2) Slow annealing:** We extend the allocation-scale annealing interval from training fractions $[0.1, 0.2]$ to $[0.1, 0.99]$, retaining the same 10% warm-up. **(3) Normalization:** We test two normalization functions, Softmax and ALN.

Results and Analysis. In Table 1, averaged over the eight configurations, the marginal accuracy gains are approximately 0.19 percentage points for two-stage selection, 0.37 points for slow annealing, and 0.10 points for ALN. These gains suggest that each component genuinely helps. Although combining two-stage selection, slow annealing, and ALN does not give the highest mean accuracy, its standard deviation is the smallest in the table. Two-stage selection can be viewed as a staged form of slow annealing, suggesting that allocating more training steps to progressively smaller budgets may help SrGS.

7 Discussion

Connection to REINFORCE. REINFORCE (**?**) learns a probability distribution over actions to maximize expected reward using sampled actions and their rewards. SrGS addresses the more specific task of selecting a subset that minimizes prediction loss under a cost budget. Viewing candidates as actions with rewards $-g_i$, subtracting the average reward yields $-U_i$ (Appendix G). This view relates SrGS's design to policy gradients: ALN preserves the relative-utility signal while removing the probability factor that suppresses updates to unlikely candidates, and UAS enables recovery from zero. Together with structural cost-aware budgeting and straight-through gradients, these mechanisms adapt stochastic selection to budgeted sparse subsets.

Limitations. Our regularizer analysis is exact only for linear models with squared loss in the low-temperature limit, whereas training uses a soft gate ($\tau = 0.1$) with a biased straight-through gradient. Empirically, SrGS is less dominant at the tightest budgets on high-dimensional data, i-Razor slightly leads on Criteo in Type-2, and the ablation covers a single dataset and budget.

Conclusion. SrGS replaces penalty tuning with structural, cost-aware budget allocation, keeps suppressed candidates recoverable via ALN, UAS, and an asymmetric clip, and, for linear models with squared loss, induces a regularizer that vanishes exactly when the weighted feature support fits within the budget. Future work will extend UAS to dynamic sparse training. See Appendix I for AI assistance details.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Dimitri P Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic press, 2014.

Chun-Hao Chang, Ladislav Rampasek, and Anna Goldenberg. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017.

Andrew Cotter, Heinrich Jiang, and Nathan Srebro. Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, 20(172):1–59, 2019.

Criteo Labs. Display advertising challenge, 2014. URL `https://www.kaggle.com/c/criteo-display-ad-challenge`.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Antonio A Ginart, Maxim Naumov, Dheevatsa Mudigere, Jiyan Yang, and James Zou. Mixed dimension embeddings with application to memory-efficient recommendation systems. In *2021 IEEE International symposium on information theory (ISIT)*, pp. 2786–2791. IEEE, 2021.

Suriya Gunasekar, Blake Woodworth, Bhojanapalli Srinadh, and Nathan Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

Samir Khuller, Anna Moss, and Joseph Seffi Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

Ismael Lemhadri, Feng Ruan, Louis Abraham, and Robert Tibshirani. Lassonet: A neural network with feature sparsity. *Journal of Machine Learning Research*, 22(127):1–29, 2021.

Yiwen Liao, Raphaël Latty, and Bin Yang. Feature selection using batch-wise attenuation and feature mask normalization. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2021.

Siyi Liu, Chen Gao, Yihong Chen, Depeng Jin, and Yong Li. Learnable embedding sizes for recommender systems. In *International Conference on Learning Representations (ICLR)*, 2021.

Shan Luo and Zehua Chen. Sequential lasso cum ebic for feature selection with ultra-high dimensional feature space. *Journal of the American Statistical Association*, 109(507):1229–1240, 2014.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

André FT Martins and Ramón Fernandez Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning (ICML)*, pp. 1614–1623, 2016.

Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

Pedram Pad, Hadi Hammoud, Mohamad Dia, Nadim Maamari, and Liza Andrea Dunbar. SAND: One-shot feature selection with additive noise distortion. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47443–47455. PMLR, 2025.

Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pp. 40–44. IEEE, 1993.

Ben Peters, Vlad Niculae, and André FT Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1504–1519, 2019.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(1): 2822–2878, 2018.

Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Steve Wang and Will Cukierski. Click-through rate prediction. `https://kaggle.com/competitions/avazu-ctr-prediction`, 2014. Kaggle.

Colin Wei, Sham Kakade, and Tengyu Ma. The implicit and explicit regularization effects of dropout. In *International Conference on Machine Learning (ICML)*, pp. 10181–10192, 2020.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10648–10659. PMLR, 2020.

Yao Yao, Bin Liu, Haoxun He, Dakui Sheng, Ke Wang, Li Xiao, and Huanhuan Cao. i-razor: A differentiable neural input razor for feature selection and dimension search in dnn-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 36(9):4736–4749, 2023.

Taisuke Yasuda, MohammadHossein Bateni, Lin Chen, Matthew Fahrbach, Gang Fu, and Vahab Mirrokni. Sequential attention for feature selection. In *International Conference on Learning Representations (ICLR)*, 2023.

A THE COMPLETE ALGORITHM

Algorithm 1 SrGS with Scale-Invariant ALN and UAS

Require: Unnormalized feature scores $t \in \mathbb{R}^N$, model parameters θ, candidate costs $c \in \mathbb{R}^N_{>0}$, target budget B, temperature τ, stability constant $\epsilon = 10^{-6}$, minibatch (X, y)

Forward pass:
Compute scale-invariant attention: $s_i = \text{ALN}(t_i) = |t_i| / \sum_k |t_k|$
Cost-aware budget scaling: $\tilde{z}_i = s_i B / \sum_k s_k c_k$ $\{\tilde{z}_i = s_i K$ when $c \equiv 1, B = K\}$
Asymmetric clip (Eq. 8): $z_i = \min\left(\tilde{z}_i + \text{sg}[\max(\tilde{z}_i, \epsilon) - \tilde{z}_i],\ 1 - \epsilon\right)$ {STE at the lower bound only}
Sample uniformly: $u_i \sim \text{Uniform}(0, 1)$
Gumbel-Sigmoid Gate: $w_i = \sigma((\text{logit}(z_i) + \text{logit}(u_i))/\tau)$
Apply mask to model inputs: $\hat{X} = X \odot w$
Evaluate task loss: $\mathcal{L} = \ell(f_\theta(\hat{X}), y)$
Backward pass:
Compute Utility Advantage via Loss Jacobian: $U_i = g_i - \bar{g}$
Pass gradients through Gumbel Gate via reparameterization trick
Pass gradients through the clip: $\partial z_i / \partial \tilde{z}_i = 1$ if $\tilde{z}_i < 1 - \epsilon$, else 0 {lower bound passes gradients; upper bound blocks them}
Pass gradients through ALN:
    If $t_i \neq 0$: $\frac{\partial \mathcal{L}}{\partial t_i} = \frac{\text{sgn}(t_i)}{Z} U_i$
    If $t_i = 0$: Invoke UAS: $\frac{\partial \mathcal{L}}{\partial t_i} = \frac{\text{sgn}(U_i)}{Z} U_i = \frac{|U_i|}{Z}$ {Escapes the origin trap}
Update t_i and θ using standard optimizer (Adam/SGD).
At convergence: return the greedy budgeted subset, i.e. rank candidates by their converged retention probability z_i (equivalently $|t_i|$) and and scan this order starting from $S = \varnothing$. For each candidate j, add it to S if $\sum_{i \in S} c_i + c_j \leq B$, otherwise skip it and continue.

B ADDITIONAL RELATED WORK

Embedding dimension search. Cost-aware selection also arises when allocating embedding dimensions under a memory budget. Mixed Dimension Embeddings (Ginart et al., 2021) assign dimensions by a power-law rule, PEP (Liu et al., 2021) learns soft thresholds that prune embedding parameters, and i-Razor (Yao et al., 2023) learns differentiable importance scores over embedding regions. SrGS instead treats this as a knapsack over embedding blocks with explicit parameter costs (Section 6.2).

Implicit regularization. Optimizers such as SGD converge to low-complexity solutions without explicit penalties (Soudry et al., 2018; Gunasekar et al., 2017), and dropout noise implicitly penalizes the trace of the Hessian, favoring flatter minima (Wei et al., 2020). Constrained formulations such as the Augmented Lagrangian method have been adapted to neural networks (Cotter et al., 2019; Bertsekas, 2014), but the penalty induced by a structural, cost-aware budget has not been characterized; Section 5 characterizes it for linear models.

C DATASETS AND EXPERIMENTAL DETAILS

C.1 CTR PREPROCESSING

Categorical features: For Avazu and Criteo, high-dimensional categorical fields are mapped to integer IDs using a frequency-based string lookup, with out-of-vocabulary (OOV) tokens handled via hash buckets. **Continuous features:** To account for the long-tail distribution of numerical values in user-log data, we apply logarithmic bucketing to Criteo integer features. They are discretized into 41 bins using the 40 boundaries $\lfloor 1.5^{i-1} - 0.49 \rfloor$, $i = 1, \ldots, 40$, before embedding lookup.

C.2 TYPE-2 PROTOCOL DETAILS

Keeping the first 4-dimension block of every field active ensures that every field is retained, excluding field removal as a source of improvement. For i-Razor, we follow its recommended non-uniform block partition and evaluate the baseline under this configuration. Field representations are zero-padded to 64 dimensions for SrGS, PEP, and i-Razor; MDE instead uses a learned linear projection from each assigned dimension to 64. Each training phase runs for 100,000 steps. During search, SrGS uses the same warm-up and annealing fractions as in Type-1. At the end of SrGS search, we reserve $4\sum_{j=1}^{F} V_j$ parameters for the always-active blocks, and the final subset is chosen by the ordinary SrGS inference rule with the remaining budget $B_{\mathrm{rem}} = B - 4\sum_{j=1}^{F} V_j$. We then instantiate a fresh embedding table of size $V_j \times d_j$ for each field and retrain from scratch.

C.3 DATASET STATISTICS

Table 2 summarizes the key characteristics of the datasets used in our experimental evaluation, covering CTR prediction and high-dimensional classification tasks with varying feature types.

Table 2: Datasets and evaluated feature budgets for Type-1.

Dataset	Classes	Features	Feature Types	Train / Test	Selected K
Avazu	2 (click)	22	Categorical	Held-out split	$5, 10, 15$
Criteo Small	2 (click)	39	13 Int, 26 Cat	Held-out split	$5, 10, 15$
ISOLET	26 (letter)	617	Continuous audio	$6{,}238/1{,}559$	$10, 20, 30, 50, 100$
DEXTER	2 (topic)	20,000	Sparse bag-of-words	$480/120$	$5, 10, 20, 50, 100$
MNIST	10 (digit)	784	Pixel intensities	$60{,}000/10{,}000$	$10, 20, 50, 100$

C.4 REPRODUCIBILITY AND HYPERPARAMETER DETAILS

We provide the optimization, randomness, evaluation, and method-specific hyperparameters used in all experiments. Method-specific hyperparameters were selected using validation performance prior to the final evaluation. After tuning, the selected hyperparameters were fixed across the five independent evaluation seeds for each configuration. All methods within a benchmark use the same data split and downstream backbone architecture.

Optimization and random seeds. We use the Adam optimizer with gradient-norm clipping (`clipnorm`=100). For ISOLET, DEXTER, MNIST, and Avazu, the learning rate is 10^{-3}; for Criteo, it is 2.282×10^{-4}. The batch sizes are 128 for ISOLET, 64 for DEXTER, 128 for MNIST, 1024 for Avazu, and 512 for Criteo. For Type-1 experiments, each selection and retraining phase uses 50,000 optimization steps on ISOLET, DEXTER, and MNIST, and 300,000 steps on Avazu and Criteo. For Type-2 experiments, both the search and retraining phases use 100,000 optimization steps.

Unless otherwise stated, every reported configuration is evaluated with five independent random seeds, $s \in \{1, 2, 3, 4, 5\}$. We report the mean and sample standard deviation across the five runs.

Evaluation protocol. The results reported in the main tables are obtained by evaluating the freshly retrained model on the full evaluation set.

SrGS configuration. SrGS uses Absolute Linear Normalization (ALN), Gumbel–Sigmoid temperature $\tau = 0.1$, and clipping constant $\epsilon = 10^{-6}$.

Type-2 embedding-dimension search. All Type-2 methods use the same CTR backbone described in Section 6.2. For Avazu, the batch size is 1024 and the learning rate is 10^{-3}; for Criteo, the batch size is 512 and the learning rate is 2.282×10^{-4}. Each search phase and each subsequent retraining phase uses 100,000 optimization steps.

Table 3: Method-specific settings for the Type-1 feature-selection experiments. Hyperparameters were selected using validation performance and then fixed across the five evaluation seeds. All methods use the same downstream backbone described in Section 6.1.

Method	Configuration used in the reported experiments
SrGS	ALN normalization; Gumbel–Sigmoid temperature $\tau = 0.1$; $\epsilon = 10^{-6}$; 10% warm-up followed by allocation annealing from 10% to 20% of training; final ranking by $\lvert t_i \rvert$.
GS	Gumbel–Softmax temperature $\tau = 1.0$; logits initialized near zero; final top-K selection by the learned feature scores.
SeqAttn	Sequential Attention with a 10% backbone warm-up before sequential feature selection.
LLY	Three-layer feature-mask network with $\mathrm{Dense}(D) \to \mathrm{Dense}(D) \to \mathrm{Dense}(D, \mathrm{sigmoid})$. Feature importance is accumulated using the exponential moving average $\mathbf{w}_{\mathrm{accum}} \leftarrow 0.9\mathbf{w}_{\mathrm{accum}} + 0.1\mathbf{m}(\mathbf{x})$, and final features are ranked by $\mathbf{w}_{\mathrm{accum}}$.
Dropout	Concrete-dropout feature ranking with temperature $\tau = 0.1$ and L_1 regularization coefficient 1.0.
STG	Gate parameters initialized to 0.5; Gaussian gate noise with standard deviation 0.5; sparsity regularization coefficient 0.1.
LassoNet	Hierarchy multiplier $M = 10$, sparsity coefficient $\lambda = 0.01$, and hierarchy penalty coefficient $\gamma = 10$. Final features are ranked by the hierarchically coupled linear and first-layer weights.
SAND	Retention logits initialized to zero, corresponding to initial retention probability 0.5; additive Gaussian distortion with standard deviation 1.0; sparsity coefficient 0.01.

Table 4: Method-specific settings for the Type-2 cost-aware embedding dimension experiments. All methods are evaluated under the same parameter budget and the common Select–Retrain protocol.

Method	Configuration used in the reported experiments
SrGS	ALN normalization; Gumbel–Sigmoid temperature $\tau = 0.1$; 10% warm-up and 10–20% budget annealing.
i-Razor	Recommended non-uniform embedding-region partition; exponential Softmax parameterization over region architecture parameters; parameter-cost regularization coefficient 10^{-5}. The cumulative-probability threshold is selected by binary search to satisfy the parameter budget, after which the discrete architecture is retrained from scratch.
MDE	Power-law dimension allocation with exponent 0.56. The global scaling factor is determined by binary search to satisfy the target parameter budget. The resulting field representations are mapped to the common 64-dimensional backbone input using learned linear projections.
PEP	Learned soft-threshold pruning with initial threshold parameter $s_{\mathrm{init}} = -10$ and scale factor $g_k = 1$. The discrete field dimensions are extracted according to the target parameter budget.

D Full Type-1 Results

Tables 5 and 6 retain the numerical results underlying Figure 1, including the CTR retraining losses. The plotted 95% confidence intervals are approximated from the reported means and standard deviations as $\bar{x} \pm t_{0.975,n-1} s/\sqrt{n}$, with $n = 5$ seeds. Intervals are shown only for SrGS.

Table 5: Type-1 feature selection on CTR datasets. AUC and Retrain Test Loss (Loss) are reported as mean ± standard deviation over five independent random seeds. Bold denotes the best reported mean for each dataset, budget, and metric.

		Avazu ($N = 22$)		Criteo ($N = 39$)	
K	Method	AUC ↑	Loss ↓	AUC ↑	Loss ↓
5	**SrGS**	$\mathbf{0.7571 \pm 0.0007}$	0.4007 ± 0.0007	$\mathbf{0.7789 \pm 0.0004}$	$\mathbf{0.4685 \pm 0.0003}$
	GS	0.7433 ± 0.0020	0.4119 ± 0.0031	0.7673 ± 0.0006	0.4779 ± 0.0005
	SeqAttn	0.7467 ± 0.0012	0.4077 ± 0.0012	0.7563 ± 0.0003	0.4858 ± 0.0003
	LLY	0.7385 ± 0.0063	0.4122 ± 0.0042	0.7497 ± 0.0077	0.4900 ± 0.0046
	Dropout	0.6963 ± 0.0000	0.4178 ± 0.0002	0.6842 ± 0.0002	0.5265 ± 0.0001
	STG	0.7335 ± 0.0028	0.4039 ± 0.0011	0.7368 ± 0.0148	0.4971 ± 0.0092
	LassoNet	0.7310 ± 0.0210	0.4171 ± 0.0086	0.7472 ± 0.0097	0.4917 ± 0.0062
	SAND	0.7502 ± 0.0001	$\mathbf{0.3956 \pm 0.0004}$	0.7659 ± 0.0007	0.4779 ± 0.0005
10	**SrGS**	$\mathbf{0.7586 \pm 0.0019}$	0.4042 ± 0.0020	$\mathbf{0.7927 \pm 0.0011}$	$\mathbf{0.4575 \pm 0.0010}$
	GS	0.7573 ± 0.0015	0.4045 ± 0.0019	0.7878 ± 0.0007	0.4615 ± 0.0006
	SeqAttn	0.7585 ± 0.0018	0.4037 ± 0.0018	0.7799 ± 0.0034	0.4683 ± 0.0027
	LLY	0.7515 ± 0.0009	0.4061 ± 0.0020	0.7684 ± 0.0074	0.4767 ± 0.0060
	Dropout	0.7377 ± 0.0003	0.4035 ± 0.0008	0.7184 ± 0.0003	0.5092 ± 0.0002
	STG	0.7474 ± 0.0084	0.3972 ± 0.0043	0.7770 ± 0.0059	0.4695 ± 0.0043
	LassoNet	0.7577 ± 0.0020	0.4043 ± 0.0033	0.7606 ± 0.0026	0.4834 ± 0.0019
	SAND	0.7581 ± 0.0059	$\mathbf{0.3922 \pm 0.0025}$	0.7827 ± 0.0054	0.4655 ± 0.0039
15	**SrGS**	$\mathbf{0.7615 \pm 0.0015}$	0.4008 ± 0.0012	$\mathbf{0.7991 \pm 0.0010}$	$\mathbf{0.4522 \pm 0.0008}$
	GS	0.7579 ± 0.0023	0.4040 ± 0.0036	0.7936 ± 0.0016	0.4573 ± 0.0012
	SeqAttn	0.7596 ± 0.0010	0.4040 ± 0.0015	0.7963 ± 0.0010	0.4548 ± 0.0008
	LLY	0.7585 ± 0.0009	0.4032 ± 0.0024	0.7903 ± 0.0020	0.4600 ± 0.0019
	Dropout	0.7604 ± 0.0005	0.3999 ± 0.0011	0.7407 ± 0.0010	0.4947 ± 0.0007
	STG	0.7610 ± 0.0024	0.3927 ± 0.0057	0.7868 ± 0.0033	0.4620 ± 0.0026
	LassoNet	0.7595 ± 0.0010	0.4030 ± 0.0011	0.7852 ± 0.0046	0.4642 ± 0.0037
	SAND	0.7611 ± 0.0043	$\mathbf{0.3907 \pm 0.0018}$	0.7934 ± 0.0006	0.4571 ± 0.0007

Table 6: Type-1 feature selection on high-dimensional datasets. All entries are mean ± standard deviation over five seeds. ISOLET and MNIST use classification accuracy; DEXTER uses AUC. Higher is better, and the best mean for each budget is in bold. A dash denotes a budget that was not evaluated.

			ISOLET (Accuracy)		
Method	$K = 10$	$K = 20$	$K = 30$	$K = 50$	$K = 100$
SrGS	0.6672 ± 0.0279	0.8467 ± 0.0066	0.8896 ± 0.0064	0.9227 ± 0.0050	$\mathbf{0.9448 \pm 0.0032}$
GS	0.6275 ± 0.0323	0.8100 ± 0.0191	0.8582 ± 0.0051	0.9014 ± 0.0036	0.9229 ± 0.0029
SeqAttn	0.6147 ± 0.0429	$\mathbf{0.8659 \pm 0.0109}$	$\mathbf{0.9086 \pm 0.0052}$	0.9292 ± 0.0027	0.9370 ± 0.0031
LLY	0.4973 ± 0.0587	0.7477 ± 0.0400	0.8474 ± 0.0462	0.9201 ± 0.0069	0.9190 ± 0.0167
Dropout	0.3249 ± 0.0039	0.3569 ± 0.0044	0.3882 ± 0.0067	0.5290 ± 0.0041	0.6354 ± 0.0047
STG	$\mathbf{0.6698 \pm 0.0115}$	0.7173 ± 0.0143	0.7882 ± 0.0231	0.8163 ± 0.0137	0.8943 ± 0.0156
LassoNet	0.4315 ± 0.0379	0.7435 ± 0.0258	0.8639 ± 0.0132	$\mathbf{0.9314 \pm 0.0049}$	0.9384 ± 0.0027
SAND	0.5993 ± 0.0361	0.8568 ± 0.0099	0.8962 ± 0.0037	0.9130 ± 0.0041	0.9366 ± 0.0026
			DEXTER (AUC)		
Method	$K = 5$	$K = 10$	$K = 20$	$K = 50$	$K = 100$
SrGS	0.7210 ± 0.1805	0.7648 ± 0.2096	0.8165 ± 0.1599	$\mathbf{0.9551 \pm 0.0125}$	$\mathbf{0.9743 \pm 0.0051}$
GS	0.5000 ± 0.0000	0.5000 ± 0.0000	0.8023 ± 0.1527	0.9070 ± 0.0339	0.8818 ± 0.0224
SeqAttn	$\mathbf{0.9160 \pm 0.0045}$	0.7428 ± 0.1985	$\mathbf{0.9339 \pm 0.0213}$	0.8483 ± 0.1745	0.9514 ± 0.0104
LLY	0.5629 ± 0.1259	0.6465 ± 0.1819	0.8136 ± 0.1571	0.8139 ± 0.1573	0.8628 ± 0.1476
Dropout	0.5117 ± 0.0067	0.4933 ± 0.0033	0.4900 ± 0.0082	0.4979 ± 0.0155	0.4698 ± 0.0148
STG	0.5558 ± 0.1117	$\mathbf{0.7876 \pm 0.0441}$	0.7434 ± 0.2007	0.8990 ± 0.0234	0.8875 ± 0.0257
LassoNet	0.5033 ± 0.0066	0.4990 ± 0.0292	0.5064 ± 0.0079	0.5307 ± 0.0653	0.6060 ± 0.0824
SAND	0.7739 ± 0.1078	0.6837 ± 0.1602	0.8317 ± 0.0414	0.9033 ± 0.0349	0.9064 ± 0.0149
			MNIST (Accuracy)		
Method	$K = 10$	$K = 20$	$K = 50$	$K = 100$	–
SrGS	0.6753 ± 0.0100	0.8751 ± 0.0067	0.9535 ± 0.0019	$\mathbf{0.9756 \pm 0.0018}$	–
GS	$\mathbf{0.7361 \pm 0.0296}$	0.8702 ± 0.0081	0.9394 ± 0.0027	0.9677 ± 0.0009	–
SeqAttn	0.6780 ± 0.0325	$\mathbf{0.8924 \pm 0.0125}$	$\mathbf{0.9600 \pm 0.0025}$	0.9709 ± 0.0021	–
LLY	0.6828 ± 0.0132	0.8313 ± 0.0216	0.9374 ± 0.0020	0.9684 ± 0.0035	–
Dropout	0.1135 ± 0.0000	0.1135 ± 0.0000	0.1178 ± 0.0001	0.2088 ± 0.0004	–
STG	0.6501 ± 0.0472	0.7952 ± 0.0191	0.9070 ± 0.0149	0.9565 ± 0.0085	–
LassoNet	0.6544 ± 0.0806	0.8224 ± 0.0275	0.9378 ± 0.0093	0.9689 ± 0.0021	–
SAND	0.7304 ± 0.0128	0.8877 ± 0.0058	0.9479 ± 0.0015	0.9620 ± 0.0016	–

E FULL TYPE-2 RESULTS

Table 7 retains the full numerical results underlying Figure 2, including losses. The plotted 95% confidence intervals are approximated as $\bar{x} \pm t_{0.975,n-1} s/\sqrt{n}$ from the reported means and standard deviations, using the completed seed count n. Intervals are shown only for SrGS when its mean is below the highest baseline mean at the same budget: Avazu at $\rho = 0.25$ and $\rho = 1.50$, and Criteo at all four budgets, each with $n = 5$.

Table 7: Type-2 embedding dimension optimization on Avazu and Criteo. AUC is mean $\pm$ standard deviation over five independent random seeds; Loss is the reported validation loss. Bold denotes the best reported mean per budget, dataset, and metric.

		Avazu		Criteo	
ρ	Method	AUC ↑	Loss ↓	AUC ↑	Loss ↓
0.25	**SrGS**	0.77206 ± 0.00044	0.38460	0.80436 ± 0.00029	0.44721
	i-Razor	$\mathbf{0.77207 \pm 0.00030}$	0.38459	$\mathbf{0.80485 \pm 0.00044}$	**0.44682**
	MDE	0.77091 ± 0.00049	0.38537	0.80436 ± 0.00024	0.44718
	PEP	0.76988 ± 0.00027	**0.38281**	0.80129 ± 0.00353	0.44736
0.50	**SrGS**	$\mathbf{0.77236 \pm 0.00008}$	0.38436	0.80505 ± 0.00033	0.44673
	i-Razor	0.77114 ± 0.00071	0.38562	$\mathbf{0.80539 \pm 0.00010}$	**0.44626**
	MDE	0.76733 ± 0.00080	0.38858	0.80496 ± 0.00048	0.44676
	PEP	0.76998 ± 0.00024	**0.38281**	0.80286 ± 0.00341	0.44630
1.00	**SrGS**	$\mathbf{0.77112 \pm 0.00069}$	0.38564	0.80560 ± 0.00030	0.44607
	i-Razor	0.76698 ± 0.00087	0.38923	$\mathbf{0.80588 \pm 0.00037}$	**0.44589**
	MDE	0.76012 ± 0.00207	0.39528	0.80498 ± 0.00033	0.44667
	PEP	0.76827 ± 0.00339	**0.38453**	0.79845 ± 0.00048	0.45027
1.50	**SrGS**	0.76620 ± 0.00156	0.38987	0.80555 ± 0.00048	0.44631
	i-Razor	0.76610 ± 0.00085	0.39021	$\mathbf{0.80583 \pm 0.00037}$	**0.44616**
	MDE	0.75550 ± 0.00153	0.40087	0.80486 ± 0.00043	0.44691
	PEP	$\mathbf{0.76989 \pm 0.00043}$	**0.38286**	0.80150 ± 0.00321	0.44733

F Deferred Proofs

F.1 Proof of the Implicit Weighted ℓ_0 Penalty

We prove Theorem 1.

Proof. Every summand in Eq. (12) is nonnegative. Write $T = \mathrm{supp}(\beta)$ and $C_T = \sum_{i\in T} c_i = \|\beta\|_{0,c}$.

($\Leftarrow$) If $C_T \le B$, choose $q_i = 1$ for $i \in T$ and $q_i = 0$ otherwise. This allocation is feasible and every summand vanishes, giving $R^*_{B,c}(\beta) = 0$.

($\Rightarrow$) Suppose $C_T > B$ and set $\gamma := \min_{i\in T} \|X_i\|_2^2 \beta_i^2 / c_i > 0$. For any feasible q with finite penalty, all q_i on T are positive, and

$$\begin{aligned} R(q;\beta) &= \sum_{i\in T} \|X_i\|_2^2 \beta_i^2 \frac{1-q_i}{q_i} \\ &\ge \sum_{i\in T} \|X_i\|_2^2 \beta_i^2 (1-q_i) \\ &\ge \gamma \sum_{i\in T} c_i(1-q_i) \ge \gamma(C_T - B) > 0. \end{aligned} \tag{15}$$

The same lower bound holds trivially when $R(q;\beta) = +\infty$. It is uniform over all feasible q, so taking the infimum gives $R^*_{B,c}(\beta) \ge \gamma(C_T - B) > 0$. □

F.2 Proof of Adaptive Squared Weighted-ℓ_1 Regularization

We prove Theorem 2.

Proof. The case $\sum_{i\in T} c_i \le B$ follows from Theorem 1, including the case $T = \emptyset$. Suppose $B < \sum_{i\in T} c_i$. A finite feasible allocation exists with all $q_i > 0$ on T. Since the objective diverges as any such q_i approaches zero, a minimizer exists and is positive on T. Increasing an unsaturated coordinate in T strictly decreases the objective. Hence every minimizer exhausts the budget and assigns zero probability outside T.

Dropping the constant term $-\|\beta\|_2^2$, the problem is to minimize $\sum_{i\in T} \beta_i^2/q_i$ subject to $\sum_{i\in T} c_i q_i \le B$ and $0 < q_i \le 1$. This is a convex problem with strictly feasible points. Its KKT conditions are

$$-\frac{\beta_i^2}{(q_i^*)^2} + \lambda c_i + \mu_i = 0, \qquad \mu_i \ge 0, \qquad \mu_i(q_i^* - 1) = 0. \tag{16}$$

Because $B < \sum_{i\in T} c_i$, at least one coordinate in T is unsaturated. On that coordinate the stationarity equation implies $\lambda > 0$. On every unsaturated coordinate, $q_i^* = |\beta_i|/\sqrt{\lambda c_i}$; on a saturated coordinate, $|\beta_i|/\sqrt{\lambda c_i} \ge 1$. This proves Eq. (13).

The unsaturated coordinates consume the residual budget $b > 0$, so

$$\sqrt{\lambda} = \frac{\|\beta_U\|_{1,\sqrt{c}}}{b}. \tag{17}$$

The saturated coordinates contribute zero variance penalty. Substituting the unsaturated allocation gives

$$\begin{aligned} R^*_{B,c}(\beta) &= \sum_{i\in U} \beta_i^2 \left(\frac{1}{q_i^*} - 1\right) \\ &= \sqrt{\lambda}\,\|\beta_U\|_{1,\sqrt{c}} - \|\beta_U\|_2^2, \end{aligned} \tag{18}$$

which yields Eq. (14). □

F.3 Proof of Lemma 1

Proof. The Softmax-weighted mean $\bar{g}$ is strictly below $\max_j g_j$ whenever utilities differ, so any maximizer has $U_j > 0$. Then for any feature i with $U_i^{(n)} > 0$, Eq. (7) gives the update stated in the lemma, which is a strict decrease because $\eta > 0$ and $s_i^{(n)} > 0$. □

F.4 Proof of Lemma 3

Proof. Since $U_i > 0$ by hypothesis, the gradient $\frac{\partial \mathcal{L}}{\partial t_i} = \frac{\operatorname{sgn}(t_i)}{Z} U_i$ shares the sign of t_i. The update $t_i^{(\text{new})} = t_i - \eta \frac{\partial \mathcal{L}}{\partial t_i}$ therefore moves t_i toward zero from either side: if $t_i > 0$ the gradient is positive and t_i decreases, and if $t_i < 0$ the gradient is negative and t_i increases. Each step changes $|t_i|$ to $\big||t_i| - a\big|$ with $a = \eta U_i / Z$. If $|t_i| > a$ this equals $|t_i| - a$; otherwise it is $a - |t_i| \leq a$, so the iterate stays within $[-a, a]$ (for the current a). □

F.5 Standard Clipping Remedies at the Origin

(A) Forward-pass clamping (e.g., `torch.clamp`). For $y = \operatorname{Clip}(v, a, b) = \min(\max(v, a), b)$, AD differentiates the active branch, so $\frac{\partial y}{\partial v} = 0$ whenever $v < a$ and $\frac{\partial \mathcal{L}}{\partial v} = \frac{\partial \mathcal{L}}{\partial y} \cdot 0 = 0$. Momentum or weight decay may still move the parameter, but the utility signal is discarded, turning the singularity into a dead-zone.

(B) In-place weight projection $P(t) = \max(t, \epsilon)$. This breaks homogeneity, since in general $s(P(c \cdot t)) \neq s(P(t))$, and sign symmetry: on the projected domain $|t_i| = t_i$, so ALN degenerates to linear normalization $s_i = t_i / \sum_k t_k$ and a suppressed parameter can no longer escape through the negative domain. It thus installs an absorbing boundary exactly where suppressed features accumulate.

F.6 Proof of the ALN Gradient Formula

We prove Lemma 2.

Proof. For $t_i \neq 0$, the derivative of the absolute value is $\operatorname{sgn}(t_i)$, so the Jacobian is $\frac{\partial s_j}{\partial t_i} = \frac{\operatorname{sgn}(t_i)}{Z}(\delta_{ij} - s_j)$. Using the chain rule, the gradient becomes:

$$\frac{\partial \mathcal{L}}{\partial t_i} = \frac{\operatorname{sgn}(t_i)}{Z} \cdot (g_i - \bar{g}) = \frac{\operatorname{sgn}(t_i)}{Z} U_i \tag{19}$$

Unlike Eq. (7), the scalar coefficient is $\frac{\operatorname{sgn}(t_i)}{Z}$, not the score s_i. Writing $Z_{-i} = \sum_{k \neq i} |t_k|$ and assuming the other features carry non-zero total magnitude ($Z_{-i} > 0$):

$$\lim_{t_i \to 0} \left| \frac{\partial \mathcal{L}}{\partial t_i} \right| = \frac{|U_i|}{Z_{-i}} > 0 \tag{20}$$

whenever $U_i \neq 0$. The attenuation factor s_i is eliminated, so the optimizer retains a signal proportional to the Utility Advantage even as $t_i \to 0$, and can therefore "see" the utility of suppressed features. □

G CONNECTION TO REINFORCE

REINFORCE (**?**) learns a distribution over actions, called a *policy*, to maximize expected reward from sampled actions and rewards. SrGS minimizes prediction loss under a subset-cost budget: in the low-temperature limit, independent gates $w_i \sim \text{Bernoulli}(z_i)$ select a subset with reward $-\mathcal{L}(w \circ \theta)$, while Stage 2 constrains z to $\mathcal{P}_{B,c}$ (Eq. 12), up to the ϵ floor.

G.1 UTILITY ADVANTAGE AND REWARD RELATIVE TO THE MEAN

A *categorical policy* chooses one candidate. Subtracting a choice-independent *baseline* preserves REINFORCE's expected gradient; with the mean reward as baseline, the difference is the *advantage*, measuring performance relative to average.

Proposition 1 (Softmax competition gradient as an expected REINFORCE gradient)**.** *For $s = \text{Softmax}(t)$, fixed losses g_i, and $\bar{g} = \sum_k s_k g_k$,*

$$\nabla_{t_i} \mathbb{E}_{k\sim s}[g_k] = \mathbb{E}_{k\sim s}[(g_k - \bar{g})\nabla_{t_i} \log s_k] = s_i(g_i - \bar{g}) = s_i U_i. \tag{21}$$

Proof. Substitute $\nabla_{t_i} \log s_k = \mathbf{1}_{\{i=k\}} - s_i$ and $\sum_k s_k(g_k - \bar{g}) = 0$. □

For rewards $Q_i = -g_i$ with mean $V = -\bar{g}$, the advantage is $A_i = Q_i - V = -U_i$. In SrGS, $g_i = \partial\mathcal{L}/\partial s_i$ is a sensitivity held fixed in the first-order approximation $\mathcal{L}(s + \Delta s) \approx \mathcal{L}(s) + \sum_i g_i \Delta s_i$. Thus the correspondence concerns the competition gradient, not the full sampler. SrGS preserves this utility signal while removing the suppressive s_i factor.

G.2 PREMATURE POLICY COLLAPSE AND ITS STRUCTURAL RESOLUTION

Premature policy collapse is the early concentration of an action distribution, which hinders the recovery of neglected actions. Under Softmax, the updates $s_i U_i$ vanish as $s_i \to 0$. Entropy-regularized variants (**?**) add a reward bonus $+\lambda_H H(s)$, equivalently loss term $-\lambda_H H(s)$, where $H(s) = -\sum_i s_i \log s_i$, favoring diffuse probabilities.

ALN instead gives $\partial\mathcal{L}/\partial t_i = \text{sgn}(t_i)U_i/Z$ for $t_i \neq 0$, with $Z = \sum_j |t_j|$ (Lemma 2). UAS enables updates at zero; asymmetric clipping preserves suppressed candidates' gradients and stops saturated candidates from absorbing probability mass. These mechanisms maintain recoverability without an entropy hyperparameter or an objective favoring diffuse selection.

G.3 PATHWISE VERSUS SCORE-FUNCTION GRADIENT ESTIMATION

For $\mathcal{J}(z) = \mathbb{E}_w[\mathcal{L}(w \circ \theta)]$, the exact derivative in z_i is the expected loss difference between including and excluding candidate i. REINFORCE's *score-function estimator* multiplies a loss minus a baseline by the log-probability derivative. For Bernoulli gates with baseline $\mathcal{L}(z \circ \theta)$, expanding at $w = z$ with fixed derivatives $d_j = \partial\mathcal{L}/\partial w_j$ gives

$$\hat{\nabla}_{z_i} \approx d_i \frac{(w_i - z_i)^2}{z_i(1 - z_i)} + \sum_{j\neq i} d_j \frac{(w_i - z_i)(w_j - z_j)}{z_i(1 - z_i)}. \tag{22}$$

The second term is *cross-talk*, noise from other gates, with variance $\sum_{j\neq i} d_j^2 z_j(1-z_j)/(z_i(1-z_i))$. This zero-mean term is uncorrelated with the first, so its variance lower-bounds that of the first-order estimator.

A *pathwise estimator* differentiates through sampled gates expressed as functions of parameters and fixed random noise. SrGS's straight-through approximation obtains per-feature derivatives in one backward pass, avoiding the explicit cross-talk term. It introduces bias, however, by replacing discrete loss differences with derivatives, and lower variance is not guaranteed in general. Sampling variance also induces the implicit regularizer characterized in Section 5. Table 8 summarizes the correspondence.

Table 8: Structural correspondence between REINFORCE (**?**) and SrGS.

Concept	REINFORCE	SrGS
Stochastic gate	Bernoulli unit $w_i \sim \text{Bernoulli}(z_i)$	Gumbel-Sigmoid + STE ($\to \text{Bernoulli}(z_i)$ as $\tau \to 0$)
Probability constraints	Unconstrained $[0,1]^N$ or simplex Δ^{N-1}	Knapsack polytope $\mathcal{P}_{B,c}$
Baseline / advantage	$A_i = Q_i - V$	Utility Advantage $U_i = g_i - \bar{g}$ (with $A_i = -U_i$)
Logit gradient (Softmax)	$\nabla_{t_i} \mathbb{E}_{k\sim s}[g_k] = s_i(g_i - \bar{g})$	$\partial\mathcal{L}/\partial t_i = s_i U_i$ (Eq. 7)
Collapse	Premature policy collapse	Softmax Vanishing Gradient Trap (Section 4.1)
Collapse remedy	Entropy bonus (loss term $-\lambda_H H(s)$)	ALN + UAS, no λ_H (Sections 4.2–4.3)
Probability bounds	–	STE at lower bound, standard clip at upper bound (Eq. 8)
Credit assignment	Sampled scalar reward (cross-talk from all other gates)	Per-feature pathwise gradient (no cross-talk term; biased)
Exploration variance	$\text{Var}(w_i) = z_i(1 - z_i)$ (noise)	Induces the weighted ℓ_0 penalty (linear models)

H Baseline Descriptions

H.1 Type-1: Feature Selection

1. **Gumbel-Softmax (GS):** A standard Gumbel-Softmax relaxation (Jang et al., 2017; Maddison et al., 2017) without our proposed self-regularization schedule. This serves as an ablation baseline.
2. **Dropout Feature Ranking (Dropout):** A variational dropout method where retention probabilities are learned subject to L_1 regularization (Chang et al., 2017).
3. **LLY:** A Group-Lasso based approach proposed by Liao et al. (2021), utilizing an auxiliary attention network for batch-wise mask attenuation.
4. **Sequential Attention (SeqAttn):** A greedy method that iteratively selects features using attention weights as a proxy for marginal gain (Yasuda et al., 2023).
5. **Stochastic Gates (STG):** Learned Gaussian-noise gates (Yamada et al., 2020) with sparsity regularization, ranking features by their gate parameters.
6. **LassoNet:** A neural selector (Lemhadri et al., 2021) with sparsity-regularized linear skip weights hierarchically linked to first-layer weights.
7. **SAND:** A one-shot selector (Pad et al., 2025) that learns feature-retention weights through additive Gaussian noise distortion.

H.2 Type-2: Embedding Dimension Optimization

1. **Mixed Dimension Embeddings (MDE):** An analytical allocation based on vocabulary cardinalities, following the power-law rule of Ginart et al. (2021) with exponent 0.56 in our experiments.
2. **Plug-in Embedding Pruning (PEP):** A pruning method that learns soft thresholds to remove redundant embedding parameters (Liu et al., 2021).
3. **i-Razor:** A differentiable method that learns the importance of embedding regions and prunes them to determine field dimensions (Yao et al., 2023).

I ACKNOWLEDGMENTS AND AI USAGE STATEMENT

We utilized a Gemini-based model as a computational and analytical aid in the development of our theoretical framework. The tool provided support in analyzing specific parameterization limits, investigating implicit regularization mechanisms, and identifying the Softmax vanishing gradient trap. While the AI assisted in exploring the UAS resolution for the computational singularity and formalizing the implicit regularizer proofs, all theoretical results were checked and validated by the human authors, who take full responsibility for them.